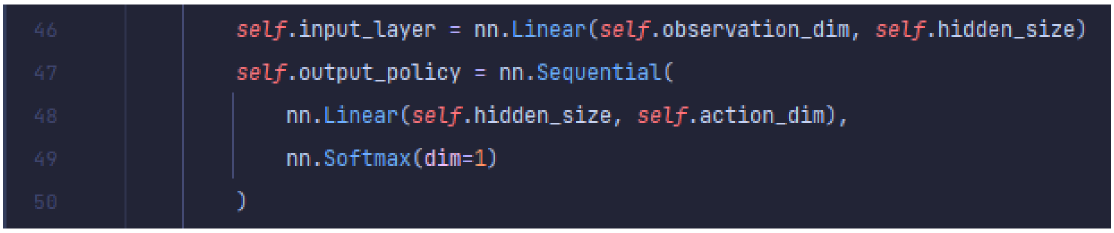
PG and VFA report

a、 In this simple cartpole problem, we don’t need to implement a complex network nor tuning a lot of parameters, it can still converge and solved easily.

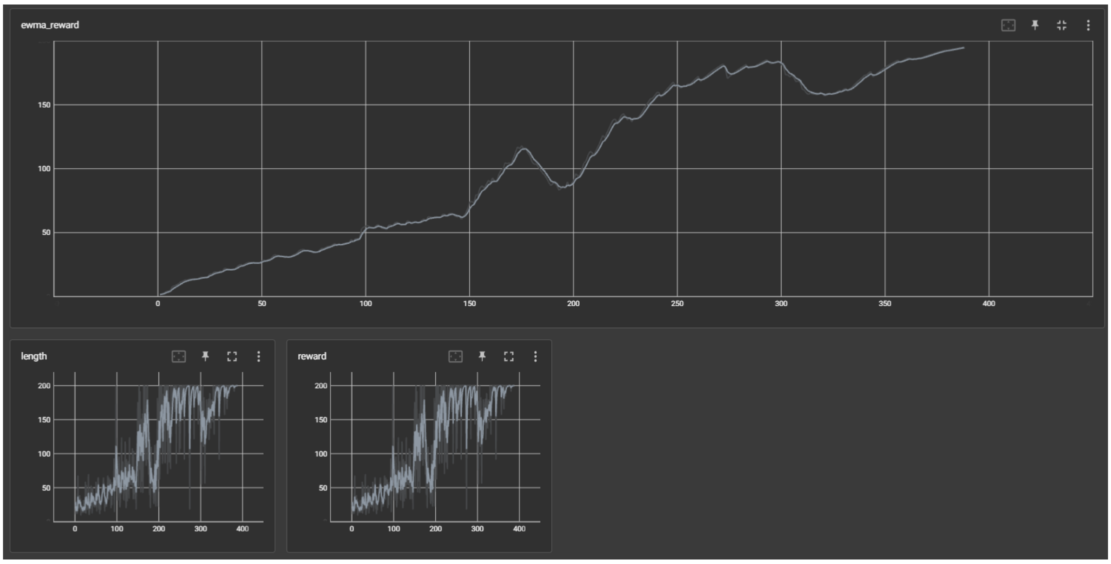
I. Network structure



```
        self.input_layer = nn.Linear(self.observation_dim, self.hidden_size)
        self.output_policy = nn.Sequential(
            nn.Linear(self.hidden_size, self.action_dim),
            nn.Softmax(dim=1)
        )
```

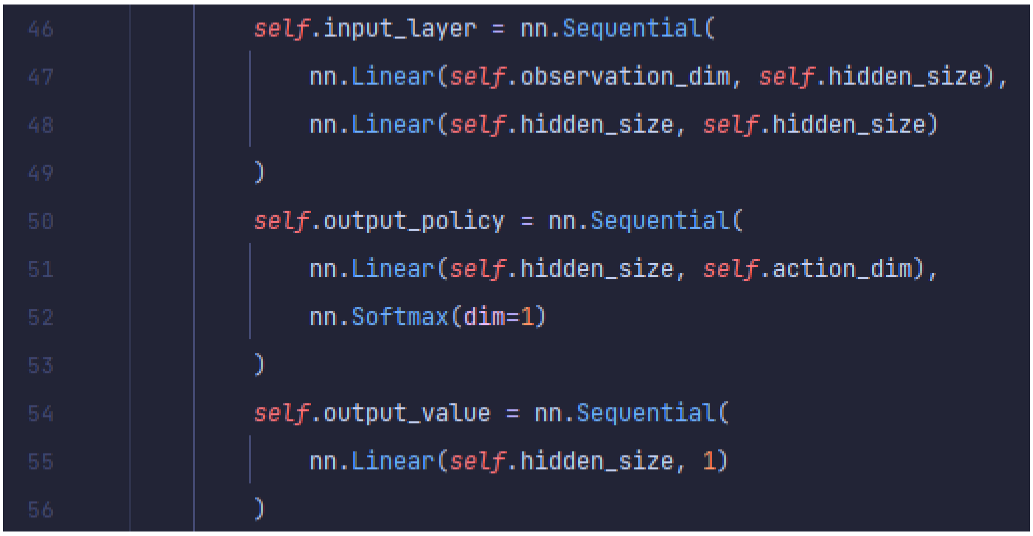
II. Hyperparameters: lr=0.01, no lr decay, gamma=0.999, seed=10

III. Tensor board



b、 I chose value function as the baseline, but it didn’t converge successfully, and I tried changing hyperparameters, using learning scheduler, changing network structure, or changing the loss function, but it still not converge. Many of the results stuck at ewma reward around -130.

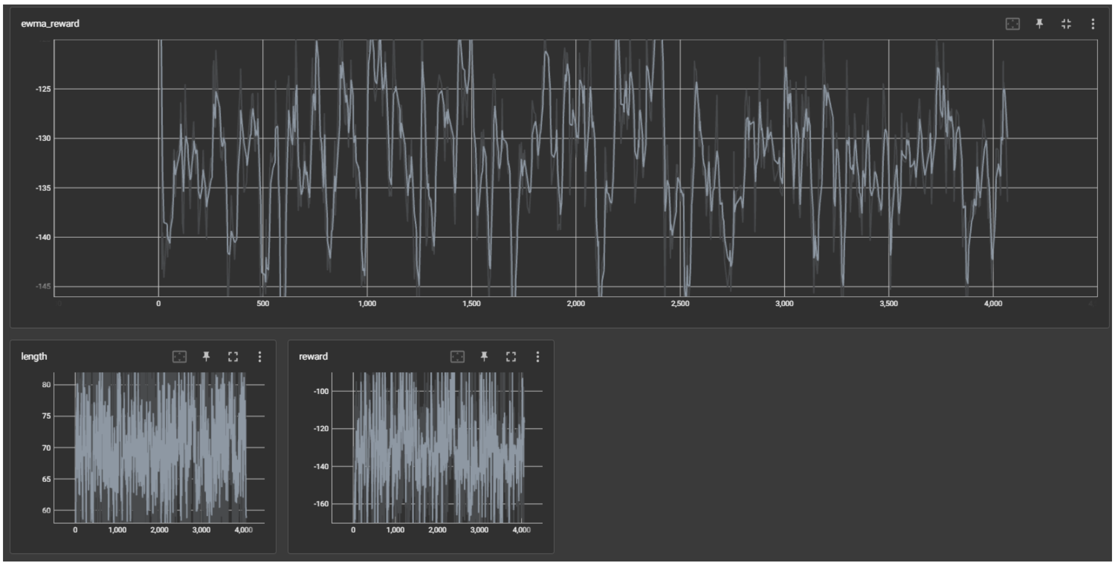
I. Network structure



```
        self.input_layer = nn.Sequential(
            nn.Linear(self.observation_dim, self.hidden_size),
            nn.Linear(self.hidden_size, self.hidden_size)
        )
        self.output_policy = nn.Sequential(
            nn.Linear(self.hidden_size, self.action_dim),
            nn.Softmax(dim=1)
        )
        self.output_value = nn.Sequential(
            nn.Linear(self.hidden_size, 1)
        )
```

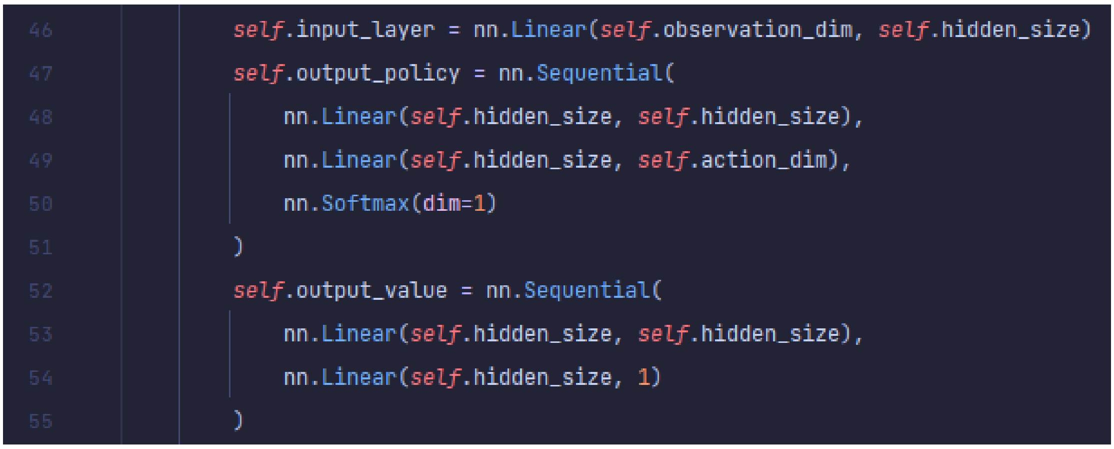
II. Hyperparameter: lr=0.0088, no lr decay, gamma=0.999, seed=88

III. Tensor board



c、 Using GAE with REINFORCE algorithm performs extremely better than the original version or with baseline. I tried lambda with 0.3, 0.7, and 0.99. I found that if lambda closer to 1, the training process converges earlier, also the reward would be more stable. When lambda=0.99 it converges at 1500 episode, when lambda=0.7 it converges at 4600 episodes, and when lambda=0.3 it not seems to converge.

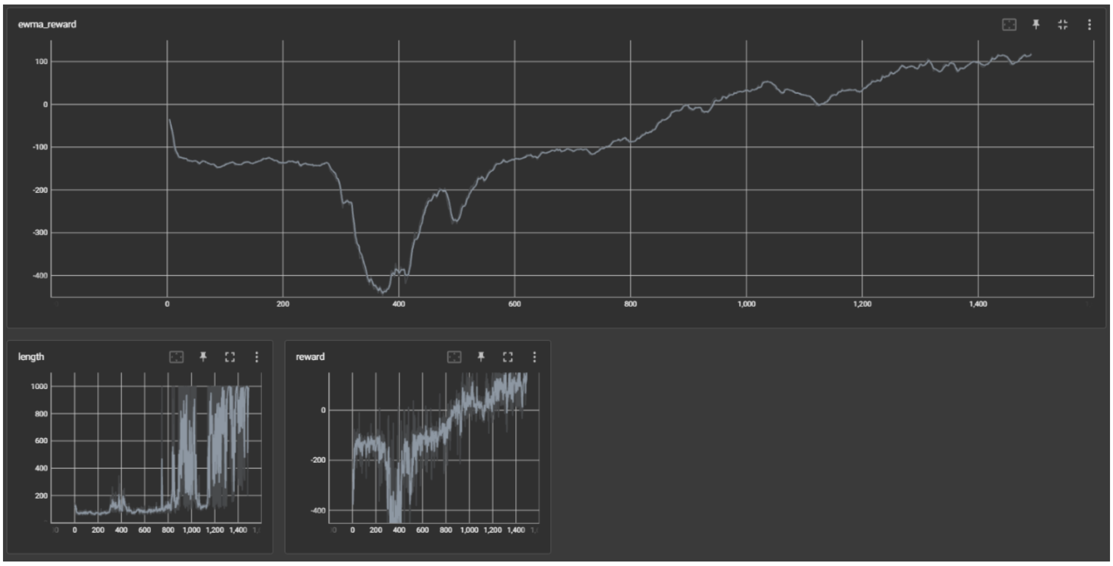
I. Network structure



```
        self.input_layer = nn.Linear(self.observation_dim, self.hidden_size)
        self.output_policy = nn.Sequential(
            nn.Linear(self.hidden_size, self.hidden_size),
            nn.Linear(self.hidden_size, self.action_dim),
            nn.Softmax(dim=1)
        )
        self.output_value = nn.Sequential(
            nn.Linear(self.hidden_size, self.hidden_size),
            nn.Linear(self.hidden_size, 1)
        )
```

II. Hyperparameters: lambda=0.99, lr=0.01, no lr decay, gamma=0.999, seed=10

III. Tensor board


ewma_reward
length
reward